

AD-A195 610 ADA (TRADE NAME) COMPILER VALIDATION SUMMARY REPORT: 1/1  
ALSYS INC ALSYCOMP 8. (U) NATIONAL COMPUTING CENTRE LTD  
MANCHESTER (ENGLAND) 19 JUN 88 AVF-VSR-90502/29

ADA (TRADE NAME) COMPILER VALIDATION SUMMARY REPORT:  
 ALSYS INC ALSYCOMP 8.. (U) NATIONAL COMPUTING CENTRE LTD  
 MANCHESTER (ENGLAND) 19 JUN 88 AVF-VSR-90502/29

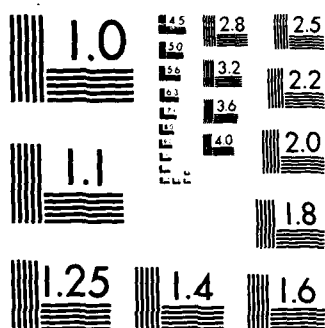
141

**UNCLASSIFIED**

F/G 12/5

■

1. 1. 1.  
 1. 1. 1.  
 1. 1. 1.  
 1. 1. 1.  
 1. 1. 1.



MICROCOPY RESOLUTION TEST CHART  
NATIONAL BUREAU OF STANDARDS-1963-A

AD-A195 618

E (When Data Entered)

## IDENTIFICATION PAGE

READ INSTRUCTIONS  
BEFORE COMPLETING FORM

ORIGINAL FILE COPY

12. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

## 4. TITLE (and Subtitle)

Ada Compiler Validation Summary Report: Alsys  
Inc., AlsysCOMP 019, V3.0.1, IBM PC AT Host,  
Intel iSBC 286/14 Target5. TYPE OF REPORT & PERIOD COVERED  
19 June 1987 to 19 June 1988

6. PERFORMING ORG. REPORT NUMBER

## 7. AUTHOR(s)

The National Computing Centre Limited  
Manchester, UK

8. CONTRACT OR GRANT NUMBER(s)

## 9. PERFORMING ORGANIZATION AND ADDRESS

The National Computing Centre Limited  
Manchester, UK10. PROGRAM ELEMENT, PROJECT, TASK  
AREA & WORK UNIT NUMBERS

## 11. CONTROLLING OFFICE NAME AND ADDRESS

Ada Joint Program Office  
United States Department of Defense  
Washington, DC 20301-3081

## 12. REPORT DATE

19 June 1987

## 13. NUMBER OF PAGES

33 p.

## 14. MONITORING AGENCY NAME &amp; ADDRESS (if different from Controlling Office)

The National Computing Centre Limited  
Manchester, UK15. SECURITY CLASS (of this report)  
UNCLASSIFIED15a. DECLASSIFICATION/DOWNGRADING  
SCHEDULE

N/A

## 16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

## 17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20. If different from Report)

UNCLASSIFIED

DTIC  
ELECTE  
JUL 12 1988  
S D

## 18. SUPPLEMENTARY NOTES

## 19. KEYWORDS (Continue on reverse side if necessary and identify by block number)

Ada Programming language, Ada Compiler Validation Summary Report, Ada  
Compiler Validation Capability, ACVC, Validation Testing, Ada  
Validation Office, AVO, Ada Validation Facility, AVF, ANSI/MIL-STD-  
1815A, Ada Joint Program Office, AJPO

## 20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

AlsysCOMP\_019, V3.1. Alsys Inc., National Computing Centre Limited, IBM PC AT under PC/DOS  
3.2 (host) and Intel iSBC 286/14 (target). ACVC 1.8.

DD

FORM

1473

EDITION OF 1 NOV 65 IS OBSOLETE

1 JAN 73

S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Ada\* COMPILER  
VALIDATION SUMMARY REPORT  
Alsys  
AlsyCOMP\_019, V3.0.1  
Host : IBM PC AT  
Target : Intel iSBC286/14

Completion of On-Site Testing  
19 June 1987

Prepared By  
The National Computing Centre Limited  
Oxford Road  
Manchester  
M1 7ED  
UK

Prepared For  
Ada Joint Program Office  
United States Department of Defense  
Washington, D.C.  
USA

---

\*Ada is a registered trademark of the United States Government  
(Ada Joint Program Office).

# Ada\* Compiler Validation Summary Report:

Compiler Name: AlsyCOMP\_019, V3.0.1

Host:

IBM PC AT under  
PC/DOS 3.2

Target:


Intel iSBC286/14

Testing Completed 18 June 1987 using ACVC 1.8

This report has been reviewed and is approved.



-----  
The National Computing Centre Ltd  
Vony Gwillim  
Oxford Road  
Manchester  
M1 7ED



-----  
Ada Validation Office  
Dr. John F. Kramer  
Institute for Defense Analyses  
Alexandria VA

*Virginia L. Castor*  
-----  
Ada Joint Program Office  
Virginia L. Castor  
Director  
Department of Defense  
Washington DC

Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	



\*Ada is a registered trademark of the United States Government  
(Ada Joint Program Office).

## EXECUTIVE SUMMARY

This Validation Summary Report (VSR) summarizes the results and conclusions of validation testing performed on the AlsYCOMP\_019, V3.0.1 using Version 1.8 of the Ada\* Compiler Validation Capability (ACVC). The AlsYCOMP\_019, V3.0.1 is hosted on an IBM PC AT operating under PC/DOS 3.2.

On-site testing was performed 15 June 1987 through 18 June 1987 at AlsYs, La Celle Saint-Cloud, France under the direction of the National Computing Centre (AVF), according to Ada Validation Organization (AVO) policies and procedures. The AVF identified 2210 of the 2399 tests in ACVC Version 1.8 to be processed during on-site testing of the compiler. The 19 tests withdrawn at the time of validation testing, as well as the 170 executable tests that make use of floating-point precision exceeding that supported by the implementation were not processed. After the 2210 tests were processed, results for Class A, C, D, or E tests were examined for correct execution. Compilation listings for Class B tests were analyzed for correct diagnosis of syntax and semantic errors. Compilation and link results of Class L tests were analyzed for correct detection of errors. There were 185 of the processed tests determined to be inapplicable; The remaining 2033 tests were passed.

The results of validation are summarized in the following table:

RESULT	CHAPTER												TOTAL
	2	3	4	5	6	7	8	9	10	11	12	14	
Passed	102	253	334	243	161	97	136	261	128	32	218	68	2033
Failed	0	0	0	0	0	0	0	0	0	0	0	0	0
Inapplicable	14	72	86	4	0	0	3	1	2	0	0	165	347
Withdrawn	0	5	5	0	0	1	1	2	4	0	1	0	19
TOTAL	116	330	425	247	161	98	140	264	134	32	219	233	2399

The AVF concludes that these results demonstrate acceptable conformity to ANSI/MIL-STD-1815A Ada.

\*Ada is a registered trademark of the United States Government (Ada Joint Program Office).

## TABLE OF CONTENTS

### CHAPTER 1 INTRODUCTION

1.1	PURPOSE OF THIS VALIDATION SUMMARY REPORT .....	1-2
1.2	USE OF THIS VALIDATION SUMMARY REPORT .....	1-2
1.3	REFERENCES .....	1-3
1.4	DEFINITION OF TERMS .....	1-3
1.5	ACVC TEST CLASSES .....	1-4

### CHAPTER 2 CONFIGURATION INFORMATION

2.1	CONFIGURATION TESTED .....	2-1
2.2	IMPLEMENTATION CHARACTERISTICS .....	2-2

### CHAPTER 3 TEST INFORMATION

3.1	TEST RESULTS .....	3-1
3.2	SUMMARY OF TEST RESULTS BY CLASS .....	3-1
3.3	SUMMARY OF TEST RESULTS BY CHAPTER .....	3-2
3.4	WITHDRAWN TESTS .....	3-2
3.5	INAPPLICABLE TESTS .....	3-2
3.6	SPLIT TESTS .....	3-5
3.7	ADDITIONAL TESTING INFORMATION .....	3-5
3.7.1	Prevalidation .....	3-5
3.7.2	Test Method .....	3-5
3.7.3	Test Site .....	3-6

### APPENDIX A COMPLIANCE STATEMENT

### APPENDIX B APPENDIX F OF THE Ada STANDARD

### APPENDIX C TEST PARAMETERS

### APPENDIX D WITHDRAWN TESTS

## CHAPTER 1

### INTRODUCTION

This Validation Summary Report <sup>(VSR)</sup> describes the extent to which a specific Ada compiler conforms to the Ada Standard. This report explains all technical terms used within it and thoroughly reports the results of testing this compiler using the Ada Compiler Validation Capability (ACVC). An Ada compiler must be implemented according to the Ada Standard and any implementation-dependent features must conform to the requirements of the Ada Standard. The Ada Standard must be implemented in its entirety, and nothing can be implemented that is not in the Standard.

Even though all validated Ada compilers conform to the Ada Standard, it must be understood that some differences do exist between implementations. The Ada Standard permits some implementation dependencies--for example, the maximum length of identifiers or the maximum values of integer types. Other differences between compilers result from characteristics of particular operating systems, hardware, or implementation strategies. All of the dependencies demonstrated during the process of testing this compiler are given in this report.

The information in this report is derived from the test results produced during validation testing. The validation process includes submitting a suite of standardized tests, the ACVC, as inputs to an Ada compiler and evaluating the results. The purpose of validating is to ensure conformity of the compiler to the Ada Standard by testing that the compiler properly implements legal language constructs and that it identifies and rejects illegal language constructs. The testing also identifies behaviour that is implementation dependent but permitted by the Ada Standard. Six classes of tests are used. These tests are designed to perform checks at compile time, at link time, and during execution.



## 1.1 PURPOSE OF THIS VALIDATION SUMMARY REPORT

This VSR documents the results of the validation testing performed on an Ada compiler. Testing was carried out for the following purposes:

- . To attempt to identify any language constructs supported by the compiler that do not conform to the Ada Standard
- . To attempt to identify any unsupported language constructs required by the Ada Standard.
- . To determine that the implementation-dependent behaviour is allowed by the Ada Standard

Testing of this compiler was conducted by NCC under the direction of the AVF according to policies and procedures established by the Ada Validation Organisation (AVO). On-site testing was conducted from 15 June 1987 through 18 June 1987 at Alsys, La Celle, Saint Cloud, France.

## 1.2 USE OF THIS VALIDATION SUMMARY REPORT

Consistent with the national laws of the originating country, the AVO may make full and free public disclosure of this report. In the United States, this is provided in accordance with the "Freedom of Information Act" (5 U.S.C. 552). The results of this validation apply only to the computers, operating systems, and compiler versions identified in this report.

The organisations represented on the signature page of this report do not represent or warrant that all statements set forth in this report are accurate and complete, or that the subject compiler has no nonconformities to the Ada Standard other than those presented. Copies of this report are available to the public from:

Ada Information Clearinghouse  
Ada Joint Program Office  
OUSDRE  
The Pentagon, Rm 3D-139 (Fern Street)  
Washington DC 20301-3081

or from:

Ada Validation Facility  
The National Computing Centre Ltd  
Oxford Road  
Manchester  
M1 7ED  
United Kingdom

## INTRODUCTION

Questions regarding this report or the validation test results should be directed to the AVF listed above or to:

Ada Validation Organization  
Institute for Defense Analyses  
1801 North Beauregard  
Alexandria VA 22311

### 1.3 REFERENCES

1. Reference Manual for the Ada Programming Language,  
ANSI/MIL-STD-1815A, FEB 1983.
2. Ada Validation Organization: Policies and Procedures, MITRE  
Corporation, JUN 1982, PB 83-110601.
3. Ada Compiler Validation Capability Implementer's Guide,  
SofTech, Inc., DEC 1984.

### 1.4 DEFINITION OF TERMS

ACVC	The Ada Compiler Validation Capability. A set of programs that evaluates the conformity of a compiler to the Ada language specification, ANSI/MIL-STD-1815A.
Ada Standard	ANSI/MIL-STD-1815A, February 1983.
Applicant	The agency requesting validation.
AVF	The National Computing Centre Ltd. In the context of this report, the AVF is responsible for conducting compiler validations according to established policies and procedures.
AVO	The Ada Validation Organization. In the context of this report, the AVO is responsible for setting procedures for compiler validations.
Compiler	A processor for the Ada language. In the context of this report, a compiler is any language processor, including cross-compilers, translators, and interpreters.
Failed test	A test for which the compiler generates a result that demonstrates nonconformity to the Ada Standard.

## INTRODUCTION

Host	The computer on which the compiler resides.
Inapplicable test	A test that uses features of the language that a compiler is not required to support or may legitimately support in a way other than the one expected by the test.
Passed test	A test for which a compiler generates the expected result.
Target	The computer for which a compiler generates code.
Test	A program that checks a compiler's conformity regarding a particular feature or features to the Ada Standard. In the context of this report, the term is used to designate a single test, which may comprise one or more files.
Withdrawn	A test found to be incorrect and not used to check conformity to test the Ada language specification. A test may be incorrect because it has an invalid test objective, fails to meet its test objective, or contains illegal or erroneous use of the language.

### 1.5 ACVC TEST CLASSES

Conformity to the Ada Standard is measured using the ACVC. The ACVC contains both legal and illegal Ada programs structured into six test classes: A, B, C, D, E, and L. The first letter of a test name identifies the class to which it belongs. Class A, C, D, and E tests are executable, and special program units are used to report their results during execution. Class B tests are expected to produce compilation errors. Class L tests are expected to produce link errors.

Class A tests check that legal Ada programs can be successfully compiled and executed. However, no checks are performed during execution to see if the test objective has been met. For example, a Class A test checks that reserved words of another language (other than those already reserved in the Ada language) are not treated as reserved words by an Ada compiler. A Class A test is passed if no errors are detected at compile time and the program executes to produce a PASSED message.

Class B tests check that a compiler detects illegal language usage. Class B tests are not executable. Each test in this class is compiled and the resulting compilation listing is examined to verify that every syntax or semantic error in the test is detected. A Class B test is passed if every illegal construct that it contains is detected by the compiler.

## INTRODUCTION

Class C tests check that legal Ada programs can be correctly compiled and executed. Each Class C test is self-checking and produces a PASSED, FAILED, or NOT APPLICABLE message indicating the result when it is executed.

Class D tests check the compilation and execution capabilities of a compiler. Since there are no requirements placed on a compiler by the Ada Standard for some parameters--for example, the number of identifiers permitted in a compilation or the number of units in a library--a compiler may refuse to compile a Class D test and still be a conforming compiler. Therefore, if a Class D test fails to compile because the capacity of the compiler is exceeded, the test is classified as inapplicable. If a Class D test compiles successfully, it is self-checking and produces a PASSED or FAILED message during execution.

Each Class E test is self-checking and produces a NOT APPLICABLE, PASSED, or FAILED message when it is compiled and executed. However, the Ada Standard permits an implementation to reject programs containing some features addressed by Class E tests during compilation. Therefore, a Class E test is passed by a compiler if it is compiled successfully and executes to produce a PASSED message, or if it is rejected by the compiler for an allowable reason.

Class L tests check that incomplete or illegal Ada programs involving multiple, separately compiled units are detected and not allowed to execute. Class L tests are compiled separately and execution is attempted. A Class L test passes if it is rejected at link time--that is, an attempt to execute the main program must generate an error message before any declarations in the main program or any units referenced by the main program are elaborated.

Two library units, the package REPORT and the procedure CHECK\_FILE, support the self-checking features of the executable tests. The package REPORT provides the mechanism by which executable tests report PASSED, FAILED, or NOT APPLICABLE results. It also provides a set of identity functions used to defeat some compiler optimization allowed by the Ada Standard that would circumvent a test objective. The procedure CHECK\_FILE is used to check the contents of text files written by some of the Class C tests for chapter 14 of the Ada Standard. The operation of these units is checked by a set of executable tests. These tests produce messages that are examined to verify that the units are operating correctly. If these units are not operating correctly, then the validation is not attempted.

## INTRODUCTION

The text of the tests in the ACVC follow conventions that are intended to ensure that the tests are reasonably portable without modification. For example, the tests make use of only the basic set of 55 characters, contain lines with a maximum length of 72 characters, use small numeric values, and place features that may not be supported by all implementations in separate tests. However, some tests contain values that require the test to be customized according to implementation-specific values--for example, an illegal file name. A list of the values used for this validation are listed in Appendix C.

A compiler must correctly process each of the tests in the suite and demonstrate conformity to the Ada Standard either meeting the pass criteria given for the test or by showing that the test is inapplicable to the implementation. Any test that was determined to contain an illegal language construct or an erroneous language construct is withdrawn from the ACVC and, therefore, is not used in testing a compiler. The tests withdrawn at the time of validation are given in Appendix D.

## CHAPTER 2

### CONFIGURATION INFORMATION

#### 2.1 CONFIGURATION TESTED

The candidate compilation system for this validation was tested under the following configuration:

Compiler: AlsyCOMP\_019, V3.0.1

ACVC Version: 1.8

Certification Expiration Date:

Host Computer:

Machine : IBM PC AT

Operating System: PC/DOS 3.2

Memory Size: 640K main memory, 8Mb extended  
memory and 30Mb hard disk

Target Computer:

Machine : Intel iSBC286/14

Operating System : Bare machine

Memory size : 2 Mbytes

Communications Network: RS232C connection with Alsys-  
provided communication software.

## 2.2 IMPLEMENTATION CHARACTERISTICS

One of the purposes of validating compilers is to determine the behaviour of a compiler in those areas of the Ada Standard that permit implementations to differ. Class D and E tests specifically check for such implementation differences. However, tests in other classes also characterize an implementation. This compiler is characterized by the following interpretations of the Ada Standard:

- . Capacities.

The compiler correctly processes compilations containing loop statements nested to 17 levels, block statements nested to 65 levels, and recursive procedures separately compiled as subunits nested to 17 levels. It correctly processes a compilation containing 723 variables in the same declarative part. (See tests D55A03A..H (8 tests), D56001B, D64005E..G (3 tests), and D29002K.)

- . Universal integer calculations.

An implementation is allowed to reject universal integer calculations having values that exceed `SYSTEM.MAX_INT`. This implementation does not reject such calculations and processes them correctly. (See tests D4A002A, D4A002B, D4A004A, and D4A004B.)

- . Predefined types.

This implementation supports the additional predefined types `SHORT_INTEGER`, `LONG_INTEGER` and `LONG_FLOAT`, in the package `STANDARD`. (See tests B86001C and B86001D.)

- . Based literals.

An implementation is allowed to reject a based literal with a value exceeding `SYSTEM.MAX_INT` during compilation, or it may raise `NUMERIC_ERROR` or `CONSTRAINT_ERROR` during execution. This implementation raises `NUMERIC_ERROR` during execution. (See test E24101A.)

## CONFIGURATION INFORMATION

### . Array Types.

An implementation is allowed to raise `NUMERIC_ERROR` or `CONSTRAINT_ERROR` for an array having a `'LENGTH` that exceeds `STANDARD.INTEGER'LAST` and/ or `SYSTEM.MAX_INT`.

A packed `BOOLEAN` array having a `'LENGTH` exceeding `INTEGER'LAST` raises no exception when the array type is declared or when the array objects are declared or sliced. (See test C52103X.)

A packed two-dimensional `BOOLEAN` array with more than `INTEGER'LAST` components raises `CONSTRAINT_ERROR` when the length of a dimension is calculated and exceeds `INTEGER'LAST`. (See test C52104Y.)

A null array with one dimension of length greater than `INTEGER'LAST` may raise `NUMERIC_ERROR` or `CONSTRAINT_ERROR` either when declared or assigned. Alternatively, an implementation may accept the declaration. However, lengths must match in array slice assignments. This implementation does not raise `NUMERIC_ERROR` when the array type is declared. (See test E52103Y.)

In assigning one-dimensional array types, the expression appears to be evaluated in its entirety before `CONSTRAINT_ERROR` is raised when checking whether the expression's subtype is compatible with the target's subtype. In assigning two-dimensional array types, the expression does not appear to be evaluated in its entirety before `CONSTRAINT_ERROR` is raised when checking whether the expression's subtype is compatible with the target's subtype. (See test C52013A.)

### . Discriminated types.

During compilation, an implementation is allowed to either accept or reject an incomplete type with discriminants that is used in an access type definition with a compatible discriminant constraint. This implementation accepts such subtype indications. (See test E38104A.)

In assigning record types with discriminants, the expression appears to be evaluated in its entirety before `CONSTRAINT_ERROR` is raised when checking whether the expression's subtype is compatible with the target's subtype. (See test C52013A.)



## CONFIGURATION INFORMATION

### . Aggregates.

In the evaluation of a multi-dimensional aggregate, all choices appear to be evaluated before checking against the index type. (See tests C43207A and C43207B.)

In the evaluation of an aggregate containing subaggregates, all choices are not evaluated before being checked for identical bounds. (See test E43212B.)

All choices are evaluated before `CONSTRAINT_ERROR` is raised if a bound in a nonnull range of a nonnull aggregate does not belong to an index subtype. (See test E43211B.)

### . Functions

An implementation may allow the declaration of a parameterless function and an enumeration literal having the same profile in the same immediate scope, or it may reject the function declaration. If it accepts the function declarations, the use of the enumeration literal's identifier denotes the function. This implementation rejects the declarations. (See test E66001D.)

### . Representation clauses.

The Ada Standard does not require an implementation to support representation clauses. If a representation clause is not supported, then the implementation must reject it. While the operation of representation clauses is not checked by Version 1.8 of the ACVC, they are used in testing other language features. This implementation accepts 'SIZE for tasks, and 'SMALL clauses; it accepts 'STORAGE\_SIZE for collections but not tasks. Enumeration representation clauses, including those that specify noncontiguous values, appear to be supported. (See tests C55B16A, C87B62A, C87B62B, C87B62C, and BC1002A.)

### . Pragmas.

The pragma `INLINE` is supported for procedures; and functions except if called within a package specification. (See tests CA3004E and CA3004F.)

### . Input/Output.

The package `SEQUENTIAL_IO` cannot be instantiated with unconstrained array types and record types with discriminants. The package `DIRECT_IO` cannot be instantiated with unconstrained array types and record types with discriminants without defaults. (See tests AE2101C, AE2101H, CE2201D, CE2201E, and CE2401D.)

## CONFIGURATION INFORMATION

### . Generics

Generic subprogram declarations and bodies can be compiled in separate compilations. (See test CA2009F)

Generic package declarations and bodies can be compiled in separate compilations. See tests CA2009C and BC3205D)

CHAPTER 3  
TEST INFORMATION

3.1 TEST RESULTS

Version 1.8 of the ACVC contains 2399 tests. When validation testing of AlsyCOMP\_003 was performed, 19 tests had been withdrawn. The remaining 2380 tests were potentially applicable to this validation. The AVF determined that 347 tests were inapplicable to this implementation, and that the 2033 applicable tests were passed by the implementation.

The AVF concludes that the testing results demonstrate acceptable conformity to the Ada Standard.

3.2 SUMMARY OF TEST RESULTS BY CLASS

RESULT	TEST CLASS						TOTAL
	A	B	C	D	E	L	
Passed	68	865	1030	13	11	46	2033
Failed	0	0	0	0	0	0	0
Inapplicable	1	2	338	4	2	0	347
Withdrawn	0	7	12	0	0	0	19
TOTAL	69	874	1380	17	13	46	2399

## 3.3 SUMMARY OF TEST RESULTS BY CHAPTER

RESULT	CHAPTER												
	2	3	4	5	6	7	8	9	10	11	12	14	TOTAL
Passed	102	254	334	243	161	97	136	261	128	32	218	68	2033
Failed	0	0	0	0	0	0	0	0	0	0	0	0	0
Inapplicable	14	72	86	4	0	0	3	1	2	0	0	165	345
Withdrawn	0	5	5	0	0	1	1	2	4	0	1	0	19
TOTAL	116	330	425	247	161	98	140	264	134	32	219	233	2399

## 3.4 WITHDRAWN TESTS

The following 19 tests were withdrawn from ACVC Version 1.8 at the time of this validation:

C32114A	C41404A	B74101B
B33203C	B45116A	C87B50A
C34018A	C48008A	C92005A
C35904A	B49006A	C940ACA
B37401A	B4A010C	CA3005A..D (4 tests)
		BC3204C

See Appendix D for the reason that each of these tests was withdrawn.

## 3.5 INAPPLICABLE TESTS

Some tests do not apply to all compilers because they make use of features that a compiler is not required by the Ada Standard to support. Others may depend on the result of another test that is either inapplicable or withdrawn. For this validation attempt, 357 tests were inapplicable for the reasons indicated:

- . C34001F and C35702A use SHORT\_FLOAT which is not supported by this compiler.
- . C87B62B uses the 'STORAGE\_SIZE clause to specify the collection size for a task type which is not supported by this compiler. The 'STORAGE\_SIZE clause is rejected during compilation.
- . D55A03E..H (4 tests) because the compiler only processes compilations containing loop statements nested to 17 levels.

# TEST INFORMATION

- . B86001D requires a predefined numeric type other than those defined by the Ada language in package STANDARD. There is no such type for this implementation.
- . C86001F redefines package SYSTEM, but TEXT\_IO is made obsolete by this new definition in this implementation and the test cannot be executed since the package REPORT is dependent on the package TEXT\_IO.
- . C96005B checks implementations for which the smallest and largest values in type DURATION are different from the smallest and largest values in DURATION's base type. This is not the case for this implementation.
- . BA2001E requires that duplicate names of subunits with a common ancestor be detected at compilation time. This compiler correctly detects the error at link time and the AVO rules that such behaviour is acceptable.
- . EA3004D requires that 3 errors be detected if pragma INLINE is supported for functions. But because this pragma has no effect when a function is called inside a package specification, only 2 of the intended errors obtain; the compiler correctly detects these two errors.
- . This implementation raises USE\_ERROR when an attempt is made to create a file. USE\_ERROR is also raised by all OPEN procedures if the NAME parameter does not identify a built-in device. As a result, the following 165 tests are inapplicable, as is CZ1103A (one of the support units), although this test does not appear in the counts.

## AE3101A

CE2102C	CE2204A..B (2 tests)	CE3104A
	CE2210A	CE3107A
CE2104A..D (4 tests)	CE2401A..F (6 tests)	CE3108A..B (2 tests)
	CE2402A.	
CE2105A	CE2404A	CE3109A
CE2106A	CE2405B	CE3110A
CE2107A..F (6 tests)	CE2406A	CE3111A..E (5 tests)
CE2108A..D (4 tests)	CE2407A	CE3112A..B (2 tests)
CE2109A	CE2408A	CE3114A..B (2 tests)
CE2110A..C (3 tests)	CE2409A	CE3115A
CE2111A..E (5 tests)	CE2410A	CE3203A
CE2111G..H (2 tests)	CE3102B	CE3208A
CE2201A..F (6 tests)	CE3103A	CE3301A..C (3 tests)
CE3302A	CE3410C..F (4 tests)	CE3704M..O (3 tests)
CE3305A	CE3411A	CE3706D,F (2 tests)
CE3402A..D (4 tests)	CE3412A	CE3804A..E (5 tests)

		TEST INFORMATION
CE3403A..C (3 tests)	CE3413A	CE3804G,I,K (3 tests)
CE3403E..F (2 tests)	CE3413C	CE3804M
CE3404A..C (3 tests)	CE3602A..D (4 tests)	CE3805A..B (2 tests)
CE3405A..D (4 tests)	CE3603A	CE3806A
CE3406A..D (4 tests)	CE3604A	CE3806D..E (2 tests)
CE3407A..C (3 tests)	CE3605A..E (5 tests)	CE3905A..C (3 tests)
CE3408A..C (3 tests)	CE3606A..B (2 tests)	CE3905L
CE3409A	CE3704A..B (2 tests)	CE3906A..C (3 tests)
CE3409C..F (4 tests)	CE3704D..F (3 tests)	CE3906E..F (2 tests)
CE3410A		

EE3102C

- . The following 170 tests make use of floating-point precision that exceeds the maximum of 15 supported by the implementation:

C24113L..Y (14 tests)  
 C35705L..Y (14 tests)  
 C35706L..Y (14 tests)  
 C35707L..Y (14 tests)  
 C35708L..Y (14 tests)  
 C35802L..Y (14 tests)  
 C45241L..Y (14 tests)  
 C45321L..Y (14 tests)  
 C45421L..Y (14 tests)  
 C45424L..Y (14 tests)  
 C45521L..Z (15 tests)  
 C45621L..Z (15 tests)

- . Also, one of the support tests, CZ1103A does not produce output equivalent to the expected output. This is because the exception USE\_ERROR is raised on all attempts to create a file within this test.

### 3.6 SPLIT TESTS

If one or more errors do not appear to have been detected in a Class B test because of compiler error recovery, then the test is split into a set of smaller tests that contain the undetected errors. These splits are then compiled and examined. The splitting process continues until all errors are detected by the compiler or until there is exactly one error per split. Any Class A, Class C, or Class E test that cannot be compiled and executed because of its size is split into a set of smaller subsets that can be processed.

Splits were required for 13 Class B tests.

B26005A	B33001A	B37004A
B43201D	B45102A	B61012A
B62001B	B62001C	B74401F
B74401R	B95069A	B95069B
		BC3205C

### 3.7 ADDITIONAL TESTING INFORMATION

#### 3.7.1 Prevalidation

Prior to validation, a set of test results for ACVC Version 1.8 produced by AlsYCOMP\_019 was submitted to the AVF by the applicant for review. Analysis of these results demonstrated that the compiler successfully passed all applicable tests, and the compiler exhibited the expected behaviour on all inapplicable tests.

#### 3.7.2 Test Method

Testing of AlsYCOMP\_019 using ACVC Version 1.8 was conducted on-site by a validation team from the AVF. The configuration consisted of two IBM PC ATs operating under PC/DOS 3.2 as host and a single iSBC286/14 as target.

A magnetic tape containing all tests was taken on site by the validation team for processing. The magnetic tape contained tests that make use of implementation-specific values were customized before being written to the magnetic tape. Tests requiring splits during the prevalidation testing were not included in their split form on the magnetic tape.

The contents of the magnetic tape were loaded first onto a VAX 780 computer. The source files were then transferred to the host computers via an Ethernet connection, where the required splits were performed.

## TEST INFORMATION

After the test files were loaded to disk, the full set of tests was compiled and linked on the IBM PC AT, and all executable tests were run. Results were compared with prevalidation results on the host computers.

The compiler was tested using command scripts provided by Alslys S.A. and reviewed by the validation team.

Tests were compiled, linked and executed (as appropriate) using two host computers and a single target computer. Test output, compilation listings, and job logs were captured on magnetic tape and archived at AVF. The listings examined on-site by the validation team were also archived.

### 3.7.3 TEST SITE

The validation team arrived at Alslys S.A. La Celle Saint-Cloud, France on 15 June 1987 and departed after testing was completed on 18 June 1987.



APPENDIX A

COMPLIANCE STATEMENT

Alsys Inc has submitted the following  
compliance statement concerning the  
AlsyCOMP\_019, V3.0.1

## COMPLIANCE STATEMENT

### Compliance Statement

#### Base Configuration:

Compiler: AlsyCOMP\_019, Version 3.1

Test Suite: Ada\* Compiler Validation Capability, Version 1.8

#### Host Computer:

Machine: IBM PC AT

Operating System: PC/DOS 3.2

#### Target Computer:

Machine: Intel iSBC286/14

Operating System: Bare machine

Communications Network: RS232 connection and Alsys-provided communications software

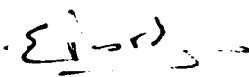
Alsys has made no deliberate extensions to the Ada language standard.

Alsys agrees to the public disclosure of this report.

Alsys agrees to comply with the Ada trademark policy, as defined by the Ada Joint Program Office.

La Gille Sr Cloud Date: June 16<sup>th</sup>, 1987

Alsys  
Etienne Morel  
Managing Director



\*Ada is registered trademark of the United States Government (Ada Joint Program Office).

## APPENDIX B

### APPENDIX F OF THE Ada STANDARD

The only allowed implementation dependencies correspond to implementation-dependent pragmas, to certain machine-dependent conventions as mentioned in chapter 13 of MIL-STD-1815A, and to certain allowed restrictions on representation classes. The implementation-dependent characteristics of the AlsyCOMP\_003, V3.0.1 are described in the following sections which discuss topics one through eight as stated in Appendix F of the Ada Language Reference Manual (ANSI/MIL-STD-1815A). (Implementation-specific portions of the package STANDARD are also included in this appendix. | The specification of the package STANDARD is also included in this appendix.)

Package STANDARD is

...

```
type INTEGER is -32768 .. 32767
type SHORT_INTEGER is range -128 .. 127 ;
type LONG_INTEGER is range -2**31..2**31-1

type FLOAT is digits 6 range -(2.0-2.0**-23)*10.0**127..
                        (2.0-2.0**-23)*10.0**127;

type LONG_FLOAT is digits 15 range
-(2.0-2.0**-51)*10.0**1023.. (2.0-2.0**-51)*10.0**1023 ;
type DURATION is delta 0.001 range -2_097_152.0..2_097_151.
999023438
```

end STANDARD;

Appendix F to the Reference Manual  
for the Ada Programming Language

Implementation dependent characteristics

The following description of the implementation dependent characteristics complies with the criteria enumerated in Appendix F of the Ada Reference Manual.

1. The following pragmas are implementation dependent:

INTERFACE  
INTERFACE\_NAME

2. The following attributes are implementation dependent:

IS\_ARRAY  
RECORD\_SIZE  
VARIANT\_INDEX  
ARRAY\_DESCRIPTOR  
RECORD\_DESCRIPTOR

3. The package SYSTEM contains what is requested by the RM. It also contains generic procedures used to perform READ/WRITE operations in memory and some generic procedures used for Bytes and Words conversion.
4. Representation clauses are supported, except for the following features:
  - (a) There is no bit implementation for any of the representation clauses.
  - (b) Machine code insertions are not implemented.
  - (c) T'SIZE is not implemented for types declared in a generic unit.
  - (d) Specification of storage for a task activation: T'SORAGE\_SIZE is not implemented when T is a task type.
  - (e) Specification of SMALL for a fixed point type: T'SMALL is restricted to a power of 2 and the absolute value of the exponent must be less than 31.
  - (f) The enumeration clause is not allowed if there a range constraint on the parent subtype.
  - (g) The record clause is not allowed for a derived record type.

5. There are 5 generated names:

IS\_ARRAY  
RECORD\_SIZE  
VARIANT\_INDEX  
ARRAY\_DESCRIPTOR  
RECORD\_DESCRIPTOR

6. Address clauses are not supported.

7. Unchecked conversions are allowed between any types.

8. Input-Output packages

Also COMP 019 raises USE\_ERROR for all CREATE procedures.  
USE\_ERROR is also raised by all OPEN procedures if the  
NAME parameter does not identify a builtin device. The  
function FORM always returns a null string.

# APPENDIX C

## TEST PARAMETERS

Certain tests in the ACVC make use of implementation-dependent values, such as the maximum length of an input line and invalid file names. A test that makes use of such values is identified by the extension .TST in its file name. Actual values to be substituted are identified by names that begin with a dollar sign. A value must be substituted for each of these names before the test is run. The values used for this validation are given below.

NAME AND MEANING	VALUE
\$BIG_ID1 Identifier the size of the maximum input line length with varying last character.	A....A1         254 characters
\$BIG_ID2 Identifier the size of the maximum input line length with varying last character.	A....A2         254 characters
\$BIG_ID3 Identifier the size of the maximum input line length with varying middle character.	A....A3A....A               127     127 characters
\$BIG_ID4 Identifier the size of the maximum input line length with varying middle character.	A....A4A....A               127     127 characters
\$BIG_INT_LIT An integer literal of value 298 with enough leading zeroes so that is is the size of the maximum line length.	0....0298         252 characters
\$BIG_REAL_LIT A real literal that can be either of floating- or fixed- point type, has value of 690.0, and has enough leading zeroes to be the size of the maximum line length.	0....069.0E1         249 characters

## TEST PARAMETER

NAME AND MEANING	VALUE
<b>\$BLANKS</b> A sequence of blanks twenty characters fewer than the size of the maximum line length.	235 spaces
<b>\$COUNT_LAST</b> A universal integer literal whose value is TEXT_IO.COUNT'LAST.	2147483647
<b>\$EXTENDED_ASCII_CHARS</b> A string literal containing all the ASCII characters with printable graphics that are not in the basic 55 Ada character set.	"abcdefghijklmnopqrstuvwxyz !\$%?@[\\]^'{}~"
<b>\$FIELD_LAST</b> A universal integer literal whose value is TEXT_IO.FIELD'LAST	255
<b>\$FILE_NAME_WITH_BAD_CHARS</b> An illegal external file name that either contains invalid characters or is too long if no invalid characters exist.	X)]!@#\$%^&~Y
<b>\$FILE_NAME_WITH_WILD_CARD_CHAR</b> An external file name that either contains a wild card character or is too long if no wild card characters exists.	XYZ*
<b>\$GREATER_THAN_DURATION</b> A universal real value that lies between DURATION'BASE'LAST and DURATION'LAST if any, otherwise any value in in the range of DURATION.	2_097_151.0
<b>\$GREATER_THAN_DURATION_BASE_LAST</b> The universal real value that is greater than DURATION'BASE'LAST, if such a value exists.	10_000_000.0
<b>\$ILLEGAL_EXTERNAL_FILE_NAME1</b> An illegal external file name.	BAD_CHARACTER*^

# TEST PARAMETERS

NAME AND MEANING	VALUE
<u>\$ILLEGAL_EXTERNAL_FILE_NAME2</u> An illegal external file name that is different from <u>\$ILLEGAL_EXTERNAL_FILE_NAME1</u> .	<u>MUCH_TOO_LONG_NAME_FOR_A_FILE</u>
<u>\$INTEGER_FIRST</u> The universal integer literal expression whose value is <u>INTEGER'FIRST</u> .	-32768
<u>\$INTEGER_LAST</u> The universal integer literal expression whose value is <u>INTEGER'LAST</u> .	32767
<u>\$LESS_THAN_DURATION</u> A universal real value that lies between <u>DURATION'BASE'FIRST</u> and <u>DURATION'FIRST</u> if any, otherwise any value in the range of <u>DURATION</u> .	-100_000.0
<u>\$LESS_THAN_DURATION_BASE_FIRST</u> The universal real value that is less than <u>DURATION'BASE'FIRST</u> , if such a value exists.	-10_000_000.0
<u>\$MAX_DIGITS</u> The universal integer literal whose value is the maximum digits supported for floating-point types.	15
<u>\$MAX_IN_LEN</u> The universal integer literal whose value is the maximum input line length permitted by the implementation.	255
<u>\$MAX_INT</u> The universal integer literal whose value is <u>SYSTEM.MAX_INT</u> .	2147483647
<u>\$NAME</u> A name of a predefined numeric type other than <u>FLOAT</u> , <u>INTEGER</u> , <u>SHORT_FLOAT</u> , <u>SHORT_INTEGER</u> , <u>LONG_FLOAT</u> , or <u>LONG_INTEGER</u> if one exists, otherwise any undefined name.	<u>LONG_LONG_INTEGER</u>



# TEST PARAMETERS

NAME AND MEANING	VALUE
<p>\$NEG_BASED_INT</p> <p>A based integer literal whose highest order nonzero bit falls in the sign bit position of the representation for SYSTEM.MAX_INT.</p>	8#77777777776#
<p>\$NON_ASCII_CHAR_TYPE</p> <p>An enumerated type definition for a character type whose literals are the identifier NON_NULL and all non ASCII characters with printable graphics.</p>	(NON_NULL)

## APPENDIX D

### WITHDRAWN TESTS

Some tests are withdrawn from the ACVC because they do not conform to the Ada Standard. The following 19 tests had been withdrawn at the time of validation testing for the reasons indicated. A reference of the form "AI-ddddd" is to an Ada Commentary.

- . C32114A: An unterminated string literal occurs at line 62.
- . B33203C: The reserved word "IS" is misspelled at line 45.
- . C34018A: The call of function G at line 114 is ambiguous in the presence of implicit conversions.
- . C35904A: The elaboration of subtype declarations SFX3 and SFX4 may raise NUMERIC\_ERROR instead of CONSTRAINT\_ERROR as expected in the test.
- . B37401A: The object declarations at lines 126 through 135 follow subprogram bodies declared in the same declarative part.
- . C41404A: The values of 'LAST and 'LENGTH are incorrect in the if statements from line 74 to the end of the test.
- . B45116A: ARRPRIBL 1 and ARRPRIBL 2 are initialized with a value of the wrong type--PRIBOOL\_TYPE instead of ARRPRIBOOL\_TYPE--at line 41.
- . C48008A: The assumption that evaluation of default initial values occurs when an exception is raised by an allocator is incorrect according to AI-00397.
- . B49006A: Object declarations at lines 41 and 50 are terminated incorrectly with colons, and end case; is missing from line 42.
- . B4A010C: The object declaration in line 18 follows a subprogram body of the same declarative part.

WITHDRAWN TESTS

- . B74101B:       The begin at line 9 causes a declarative part to be treated as a sequence of statements.
- . C87B50A:       The call of "/"= at line 31 requires a use clause for package A.
- . C92005A:       The "/"= for type PACK.BIG\_INT at line 40 is not visible without a use clause for the package PACK.
- . C940ACA:       The assumption that allocated task TT1 will run prior to the main program, and thus assign SPYNUMB the value checked for by the main program, is erroneous.
- . CA3005A..D:     No valid elaboration order exists for these tests.  
  (4 tests)
- . BC3204C:       The body of BC3204C0 is missing.

END

DATE

FILMED

9-88

DTIC